P170M109 Computational Intelligence and Decision Making

Introduction

assoc. prof. dr. Dalia Čalnerytė
assoc. prof. dr. Andrius Kriščiūnas
assoc. prof. dr. Agnė Paulauskaitė-Tarasevičienė

2022

Introduce yourself

1. What is your background (bachelor in …) ?
2. What is your area of interest (master’s thesis in …)?
3. What do you expect to learn in this course?
4. What is your experience in working with artificial intelligence?

The course covers:

1. **Computer Intelligent paradigms and Decision-making theory**
2. **Optimization and Search**
   - 2.1 Evolutionary learning
   - 2.2 Gradient optimization methods
3. **Supervised learning**
   - 3.1 Decision Trees
   - 3.2 K nearest neighbours
   - 3.3 Probability based learning
   - 3.4 The multi-Layer perceptron and radial basis functions
   - 3.5 Convolutional Neural Network
4. **Unsupervised learning**
   - 4.1 K-means method
   - 4.2 Fuzzy C-means method and fuzzy logic
   - 4.3 Self-organizing map
5. **Reinforcement learning**
   - 5.1 Markov decision process
   - 5.2 Q-learning
   - 5.3 Monte-Carlo tree search

Assessments

LD1 (20%): Input and Output analysis / Supervised learning (KNN / Decision tree / Random forest) (5w)

LD2 (10%): Unsupervised learning (k-means / fuzzy c-means) (8w)

LD3 (15%): Reinforcement learning (MCTS, …) (13w)

Problem-solving task (25%)

Exam (30%)

LD1. Input output analysis / KNN / Decision tree / Random forest

Problem: based on the given data of historical real estate transactions create the decision-making model (DMM) which aims to predict prices of new real estate objects.

Project workflow:

P1. Perform given data analysis and preprocessing
P2. Implement K-Nearest Neighbors (KNN), Decision tree (DT), and random forest (RF) algorithms (You cannot use library functions for these algorithms)
P3. Use implemented algorithms to create DMM for the given problem and evaluate the results.
P4. Use "scikit-learn" (or other) library functions for the same algorithms and evaluate the results.
P5. Write conclusions.

Data:

- **historicalData.tsv** – data to create the DMMs.
- **newData.tsv** – assume, that you don't have this data. We use this data only to evaluate (during work defense) how DMM model works with unseen data.

Data based on: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

LD1. Input output analysis / KNN / Decision tree / Random forest

***P1*. Data analysis and preprocessing**

- determine data types of features.
- provide data quality report for all features (analyze categorical and continuous features separately). The form of the report is given in slides (Input Analysis).
- provide distribution characteristics (histogram, frequency table, bar plot, box plot, pie chart …) based on the data type for each feature. For numerical features, explore the shape of distribution, perform standardization or normalization. Consider the normality of data.
- comment whether it is possible to include derived features (ratios, flags, mapping, etc.) and append Analytics Base Table (ABT) if needed.
- perform data preprocessing actions if You think it is necessary

***P2*. Implementation of KNN, DT, and RF**

- Requirements for algorithms (parameters to change)
  - KNN – k value
  - DT – minSamplesLeaf; maxDepth.
  - RF – nEstimators;
- Implementation can be based on lectures or online materials, but references to the original source (if used) are necessary. Student must be able to comment any line (even if it is used from examples or other materials).

P3 and P4.

- Creating DMM means to select optimal hyper-parameters and perform output analysis.
- Evaluate results using MAE, MAPE metrics.
- Scikit-learn library models:
  - https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
  - https://scikit-learn.org/stable/modules/tree.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

***P5*. Work conclusions**

- compare the results of DMMs that use Your implementation and “scikit-learn” library.

LD2. Unsupervised learning (k-means / fuzzy c-means)

Example 1

Data: https://osp.stat.gov.lt/statistiniu-rodikliu-analize#/

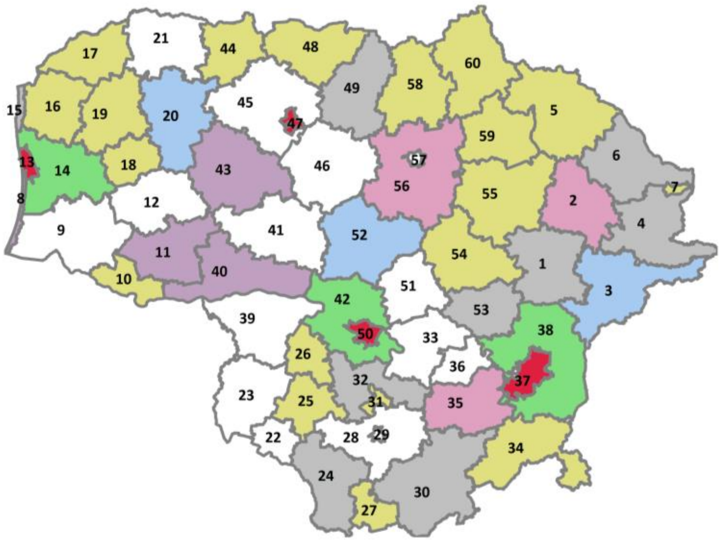
Demographic indicators:

population size, population density, ageing index, births and deaths ...

Additional indicators:

newcomers, emigrants, immigrants, departures, net migration, net inside migration ...

Results example:



LD2. Unsupervised learning (k-means / fuzzy c-means)

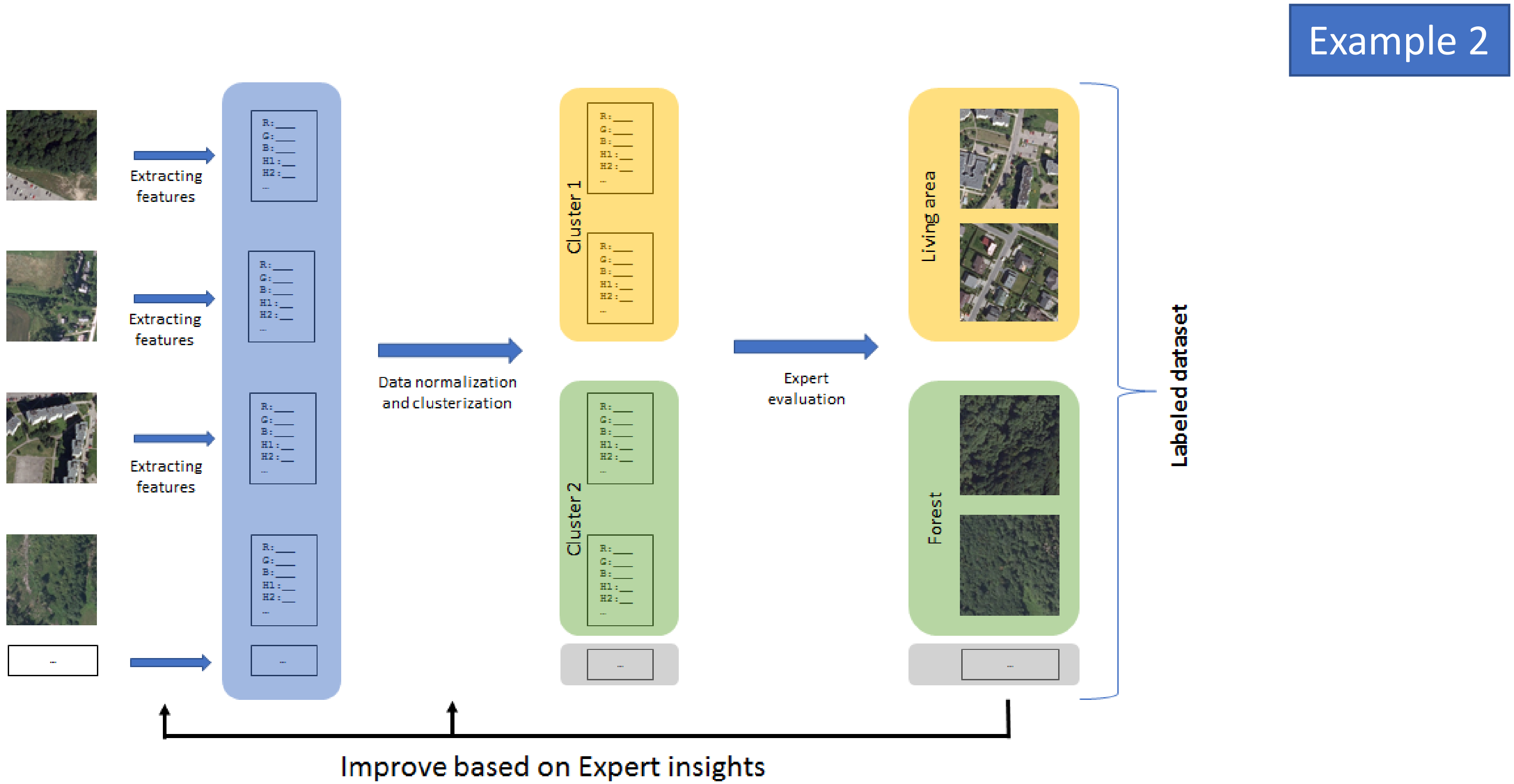
Example 2

Small size (size should be selected under expert evaluation) image dataset preparation

....

....

LD2. Unsupervised learning (k-means / fuzzy c-means)



LD3. Reinforcement learning

Example 1

Use Reinforcement learning methods to train a player in a card game (Blackjack with non-standard rules applied). The objective is to obtain cards the sum of whose is as great as possible without exceeding 21.

RULES:

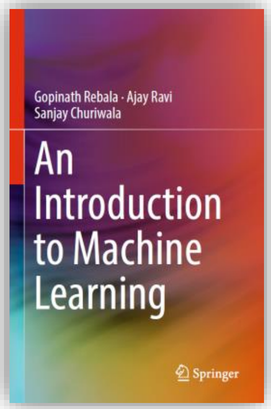
- The game is played with infinite deck of cards (cards are sampled with replacement).
- Each draw from the deck results in a value between 1 and 10 (uniformly distributed) with a color of red (♥♦, probability 1/4) or black (♠♣, probability 3/4). No face cards are used in this game, the value of ace card is 1. The values of the black cards are added, the values of the red cards are subtracted.
- At the start of the game the player and the dealer draw one black card (fully observed).
- The game is played in turns, that is, the player asks for another card one by one (***hits***) until decides to stop (***sticks***) or ***goes bust*** (the sum of card values exceeds 21 or is less than 1) and loses the game immediately. If the player sticks, it is the dealer's turn to play. The dealer plays according to the predefined rules, that is, sticks if the sum is 17 or greater and hits otherwise.
- After the dealer finishes the turn, the result of the game (wining, losing, drawing is decided):
  - The player wins if either:
    - The dealer goes bust (exceeds 21 or collects cards with the sum less than 1);
    - The sum of the card values is closer to 21 than the sum of the dealer's cards.
  - If both player and dealer have collected cards of the same sum, the game results in a draw.
  - Otherwise, the player loses.

Literature

Material on Moodle

- Lecture notes with references to external sources
- Self assessment tests
- Code examples

Rebala, Gopinath, Ajay Ravi, and Sanjay Churiwala. *An Introduction to Machine Learning*. Springer, 2019.



https://link.springer.com/book/10.1007%2F978-3-030-15729-6

Use KTU VPN or perform search through https://vb.ktu.edu (uses SSO login and proxy to access full text document)

What is AI?

5 min MOODLE test

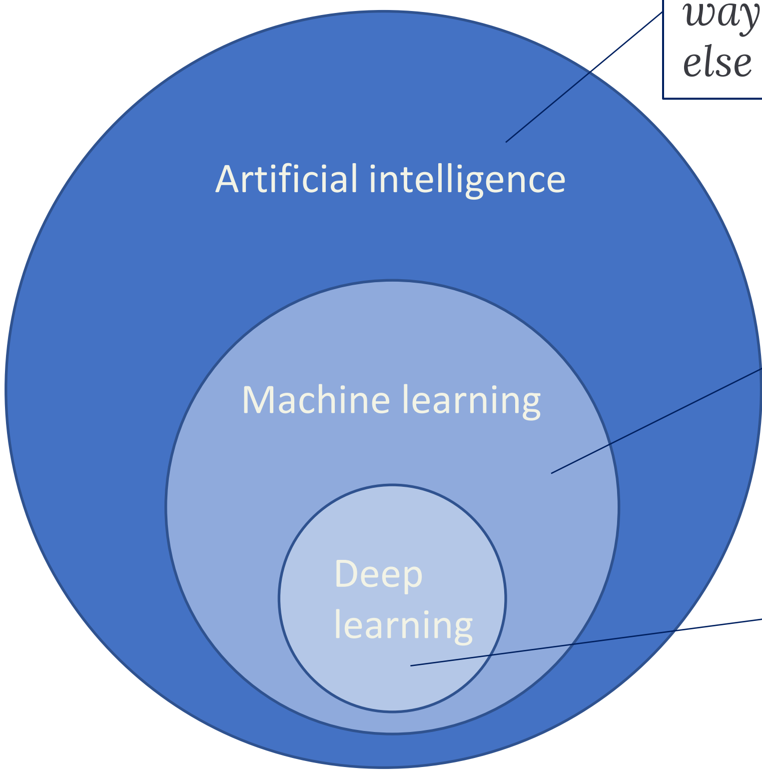
What is AI?

Artificial intelligence – systems that demonstrate intelligent behavior by analyzing its environment and making fairly independent decisions to achieve a goal.

AI types:

- **Weak AI** (*narrow AI*) has a narrow scope of functions, simulates human behavior, focuses on doing one task really well.
- **Strong AI** (*general AI*) exhibits human-level intelligence.
- **Super AI** (*superintelligence*) surpasses human intelligence and ability.

What is AI?



AI enables the machine to behave in a specific way in certain situations; it can use logic, if-else rules, decision trees, machine learning.

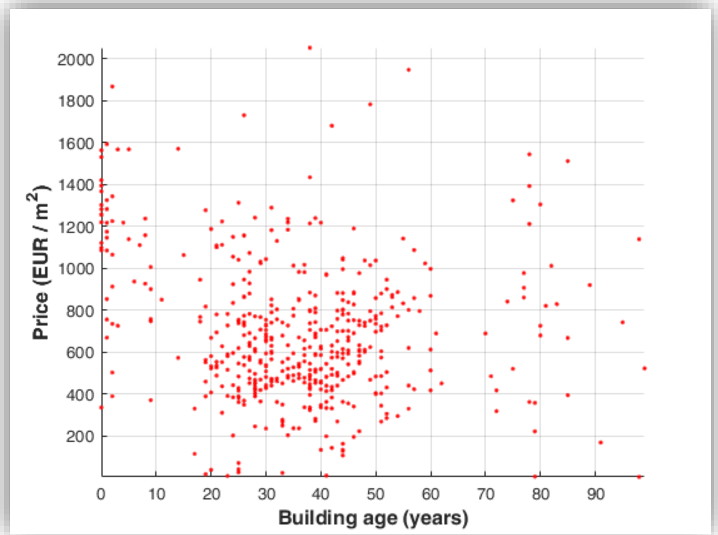
Training machines to get better at a task with experience without explicit programming.

Training machines by using multilayered neural networks.

Problem examples

Example No. 1 Regression problem

Let's say, we have data about sold flats and want to obtain approximating function, which describes how approximately price depends on the house building age



Example question: what is the flat price of building of 40 years ago?

We already know how to apply least square approximation for this problem.

Example No. 1 Regression problem

Let's say the predicted value is $\boldsymbol{f(x) = \theta_0 x^0 + \theta_1 x^1 + \cdots + \theta_n x^n}$

If we have $\boldsymbol{m}$ points, the total error (cost function) over all points can be calculated as: $$\boldsymbol{J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (f(x_i) - y_i)^2}$$

$\boldsymbol{y_i}$ actual value respectively

$\boldsymbol{f(x_i) - y_i}$ is difference between predicted and actual value

$\boldsymbol{m}$ is number of data points in dataset

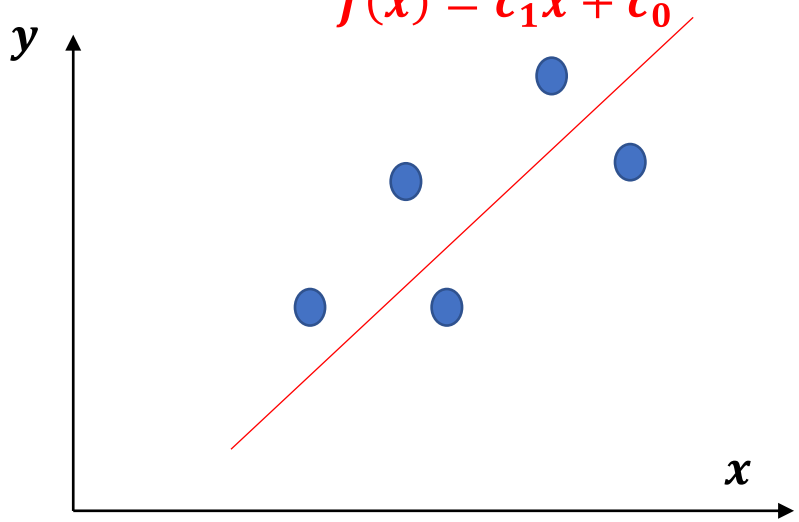
$\boldsymbol{J(\theta)}$ - total error

$\boldsymbol{\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}}$ - unknown coefficients

Example No. 1 Regression problem

Given points: $(x_i, y_i), i = 1 \dots m$

Objective: $\boldsymbol{f(x) = ?}$



The approximation quality estimation function

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(f(x_i) - y_i)^2$$

Objective: $\min_{\theta} J(\theta)$

Example No. 1

Regression coefficients are obtained analytically solving system of linear equations

Given points: $\left(x_i, y_i\right), \quad i = 1, \dots, n$

Linear combination of selected base functions with weighted coefficients

$$f(x) = \left[g_1(x) \quad g_2(x) \quad \dots \quad g_{m-1}(x) \quad g_m(x) \right] \begin{Bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{m-1} \\ c_m \end{Bmatrix} = \left[\mathbf{g}(x) \right] \left\{ \mathbf{c} \right\}$$

coefficients

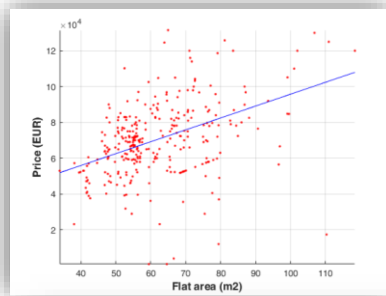
Solving linear equation system by any method (i.e. Gaussian elimination, LU decomposition, etc.)

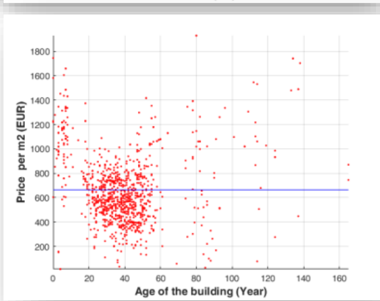
$$\left(\left(\mathbf{G}^T \right)_{m \times n} \mathbf{G}_{n \times m} \right)_{m \times m} \mathbf{c}_{m \times 1} = \left(\mathbf{G}^T \right)_{m \times n} \mathbf{y}_{n \times 1}$$

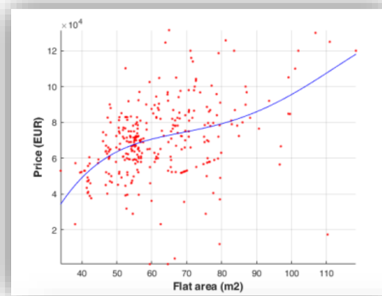
Example No. 1

$$f(x) = c_0 + c_1 x$$

$$f(x) = c_0 + c_1 x + \cdots + c_5 x^5$$







Question: what form of regression function should be selected?

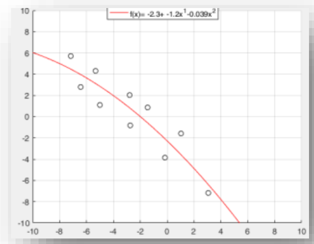
Linear regression

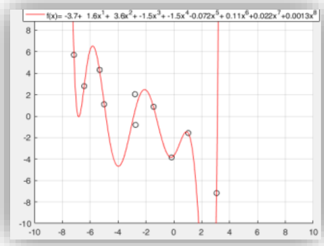
Non-linear regression

Example No. 1

The approximation quality estimation function

$$\Psi = \frac{1}{2}\sum_{j=1}^{n}\left(f\left(x_j\right) - y_j \right)^2$$





In order to avoid "**overfitting**", data should be split to different datasets:

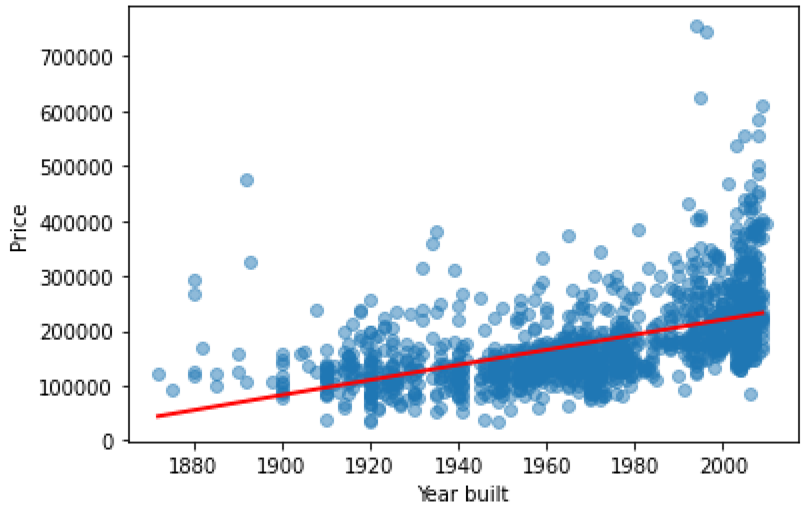
"**Training data**" – to obtain the approximating function

"**Validation data**" - to validate the approximating function

Example No. 1

Code example:

- regressionExample.ipynb
- continuosDataExample.tsv



To Do:

1. Try different polynomial functions to analyze the results
2. Search and select built in python functions and compare results (built in functions with manually obtained values)

Example No. 2.

Input

Output

X1	X2	X3	Y
0,22	23	2	100
0,35	45	4	123
0,87	76	8	233
0,99	99	14	278
0,67	45	5	134
		…	…
21	94	21	309
3	19	3	89

$\boldsymbol{y} = ?$

Example No. 2. Optimization problem

The target is to find such coefficients $\boldsymbol{\theta} = \{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \dots, \boldsymbol{\theta_n}\}$ that total error $\boldsymbol{J}(\boldsymbol{\theta})$ of selected problem be as small as possible.

The optimization problem can be formulated as:

Scalar **Objective Function**

$$\min_{\boldsymbol{\theta}} \boldsymbol{J}(\boldsymbol{\theta})$$

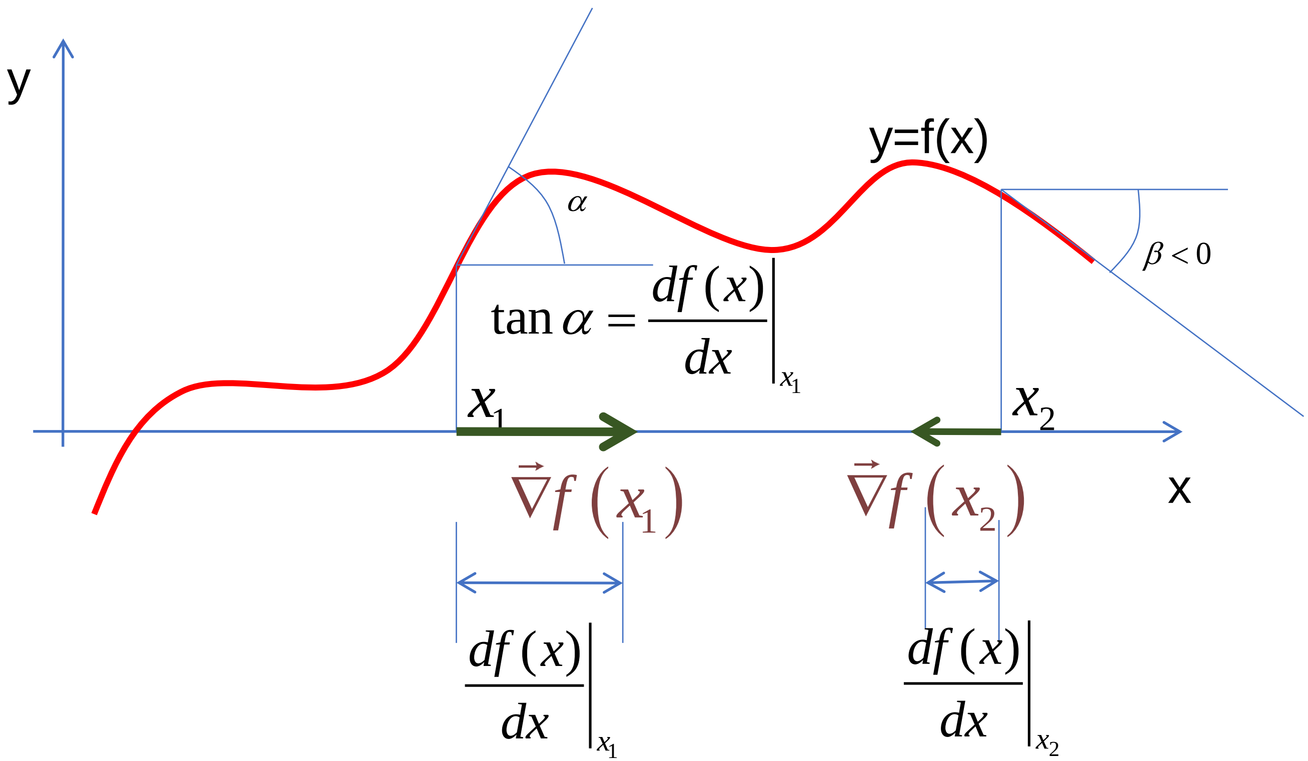
Argument (vector)

Function $\boldsymbol{J}$ minimum it's lowest value from all available values;

The minimization problems refer to "*find the values of argument vector* $\boldsymbol{\theta}$ *which provide the lowest value for function* $\boldsymbol{J}$";

If all values of vector $\boldsymbol{\theta}$ are available, we have unconstrained optimization problem

Example No. 2. Gradient explanation



Example No. 2. Gradient explanation

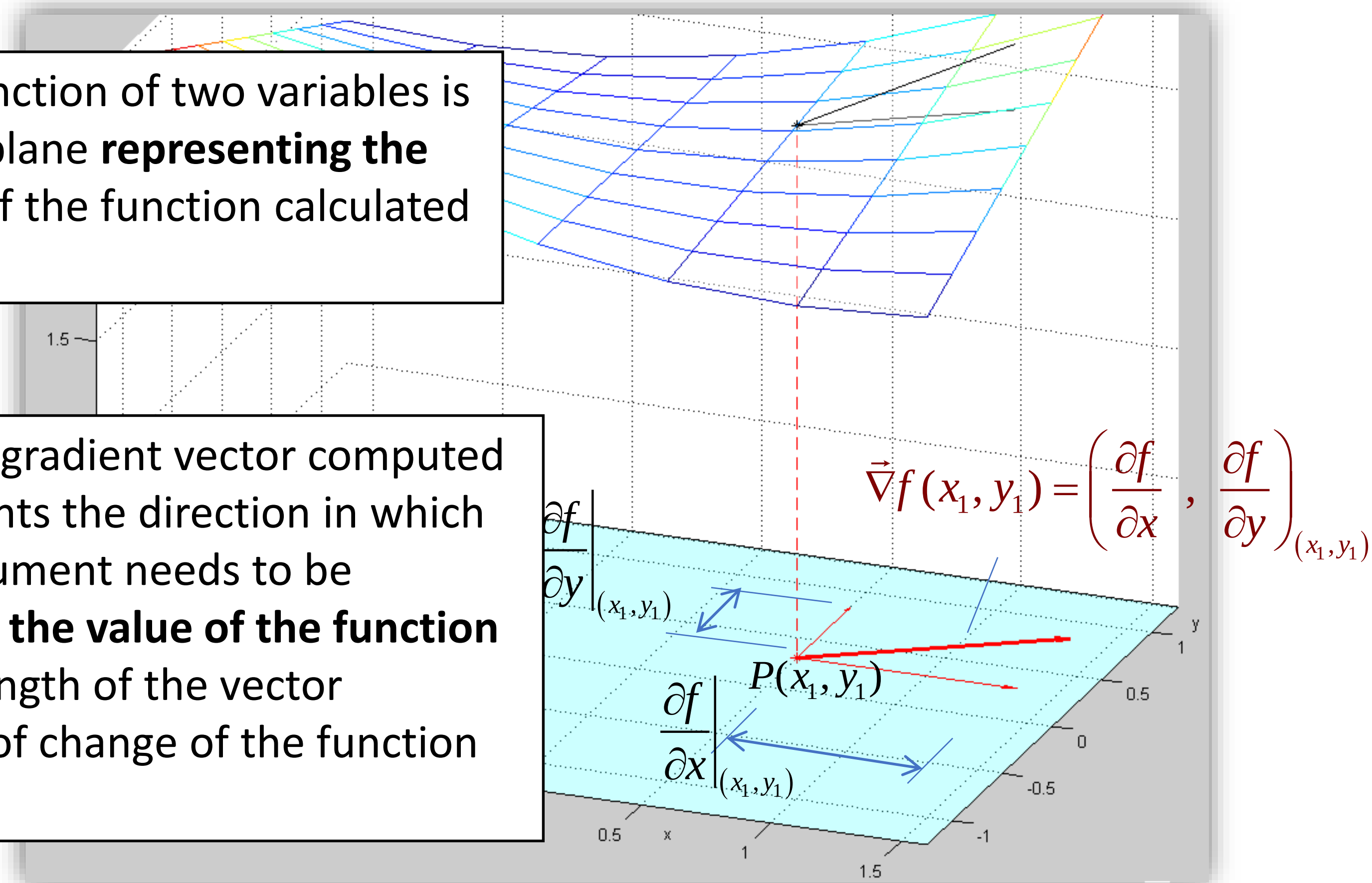
1. A function gradient is a **vector representing a derivative** of a function calculated at a given point

2. The direction of the gradient vector calculated at the point represents the direction in which the value of the argument **needs to be changed to increase the value** of the function.

3. The length of the vector represents the rate of change of the function at that point

Example No. 2. Gradient explanation

The gradient of a function of two variables is a vector in the xOy plane **representing the partial derivatives** of the function calculated at a given point



The direction of the gradient vector computed at the point represents the direction in which the value of the argument needs to be changed **to increase the value of the function most rapidly**. The length of the vector represents the rate of change of the function at that point

Example No. 2. Gradient explanation

$$\vec{\mathbf{g}} = \vec{\nabla} f(x_1, y_1) \Big/ \left\| \vec{\nabla} f(x_1, y_1) \right\|$$

- The length of the gradient vector is measured **in different units** than the values of the function and its arguments. Therefore, the scales of the function and its gradient representation are different in the same drawing;

- The **unit gradient vector** is used to determine the direction of the fastest change in the function

Example No. 2. Gradient Descend and Conjugate gradient methods

Gradient descend: in each step, changes in the arguments obtained in the opposite direction of the gradient

1. Obtain gradient $$\nabla \mathbf{J} = \left(\frac{\partial \mathbf{J}}{\partial \boldsymbol{\theta}_1}, \frac{\partial \mathbf{J}}{\partial \boldsymbol{\theta}_2}, \dots, \frac{\partial \mathbf{J}}{\partial \boldsymbol{\theta}_n}\right)$$

learning rate

2. Obtain new parameter values $$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \boxed{\Delta \boldsymbol{s}} \cdot \nabla \mathbf{J}$$

If value of target function grows, decrease step or ending the optimization

Conjugate gradient: after calculating the gradient vector, it is moved in the opposite direction until the function continues to decrease

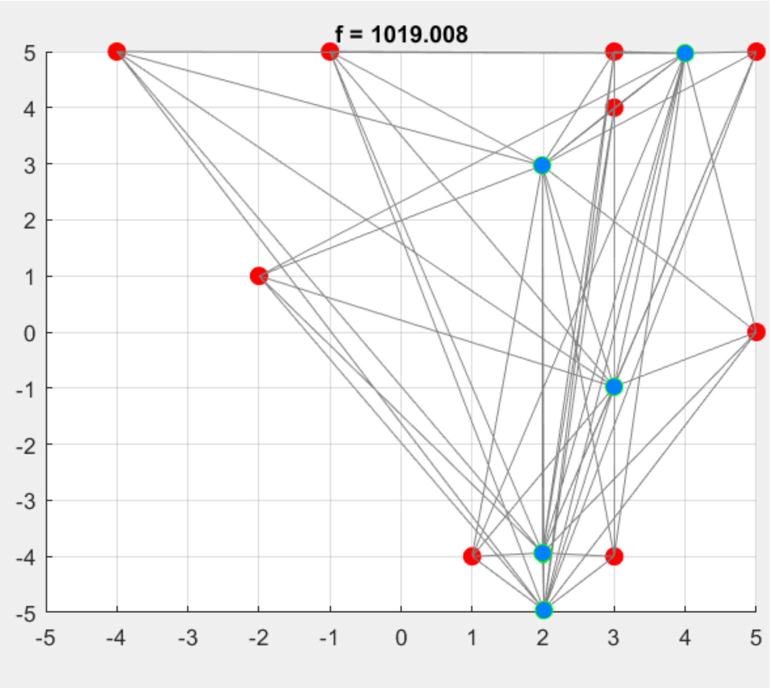
Example No. 2. Obtaining Derivatives Numerically

$$\nabla \boldsymbol{J} = \left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\theta}_1}, \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\theta}_2}, \dots, \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\theta}_n} \right)$$

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{\theta}_i} = \frac{\boldsymbol{J}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_i + h, \dots, \boldsymbol{\theta}_n) - \boldsymbol{J}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_i, \dots, \boldsymbol{\theta}_n)}{h}$$

```
function quasiGrad(data, y, W, i, currTarget, targetFnc)
    h = 0.0001;
    W(i) = W(i) + h;
    grad = (targetFnc(data, y, W) - currTarget) / h;
    return grad;
end
```

Example No. 2. Other Gradient optimization Applications



Initial data: list of **N** “fixed points”

Problem: where **M** additional points should be added, that lengths of lines between initial and added points are most uniform?

$$\min_{x_1,\dots,x_M,y_1,\dots,y_M} \Psi = \sum_{i=1}^{M} \sum_{j=i+1}^{N} \left(\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 - \bar{d} \right)^2$$

Example No. 2.

Input

Output

X1	X2	X3	Y
0,22	23	2	100
0,35	45	4	123
0,87	76	8	233
0,99	99	14	278
0,67	45	5	134
		...	...
21	94	21	309
3	19	3	89

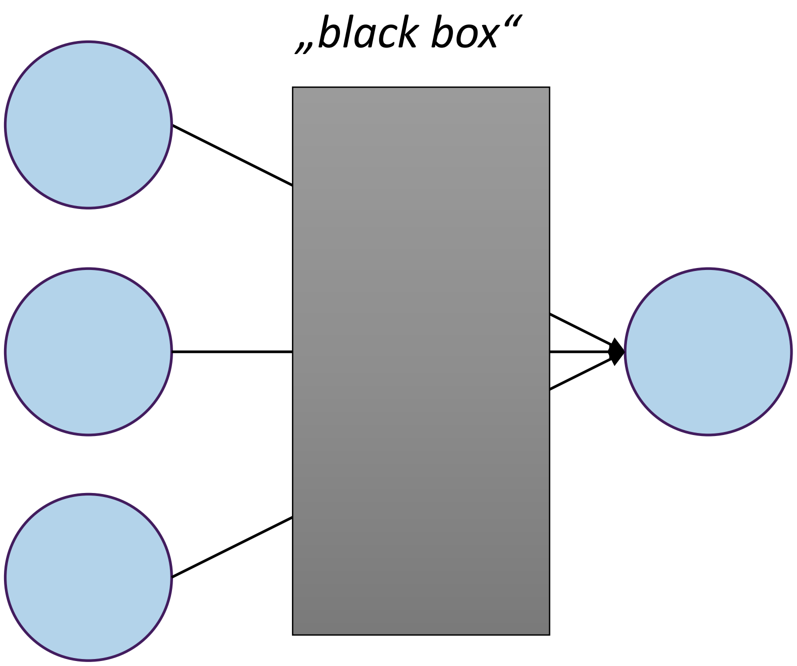
$$y = ?$$

Example No. 2 Artificial Neural Network

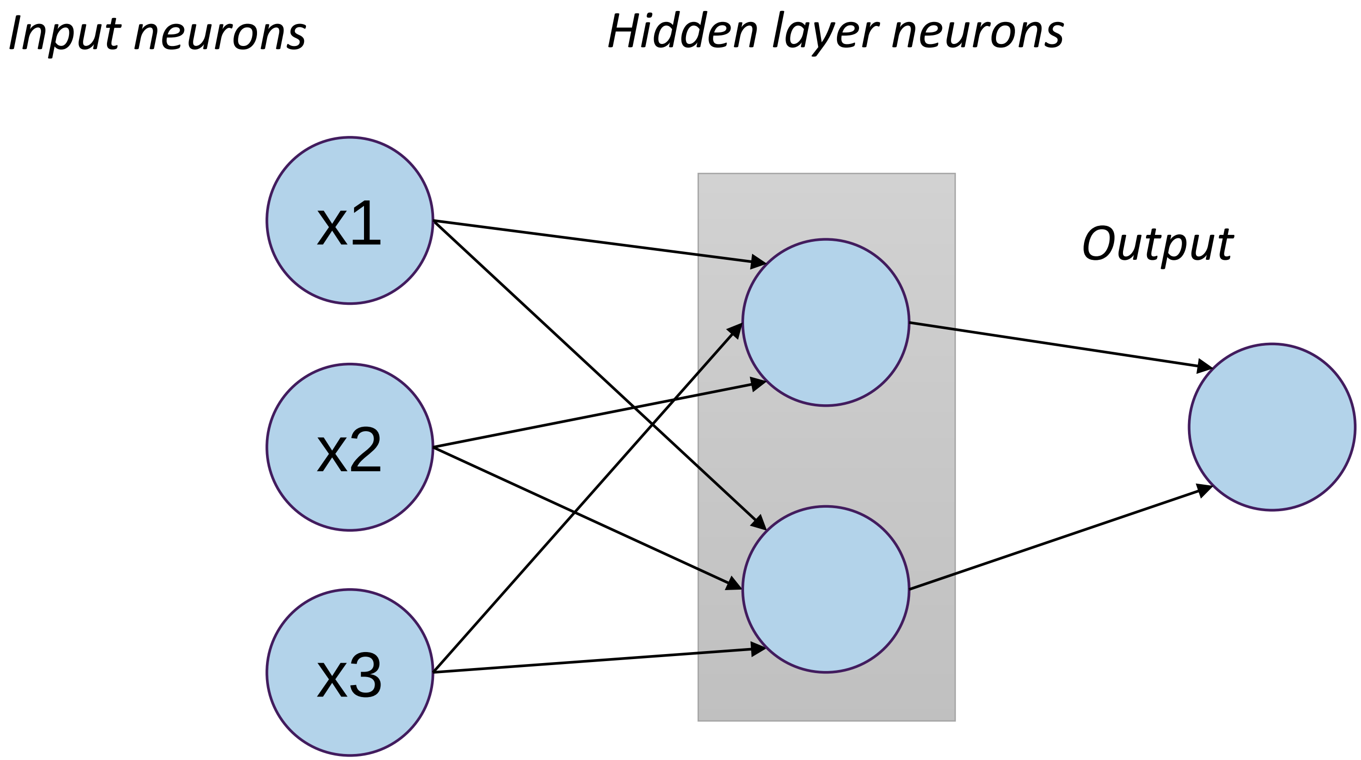
Input

Output

X1	X2	X3	Y
0,22	23	2	100
0,35	45	4	123
0,87	76	8	233
0,99	99	14	278
0,67	45	5	134
		…	…
21	94	21	309
3	19	3	89



Example No. 2 Artificial Neural network



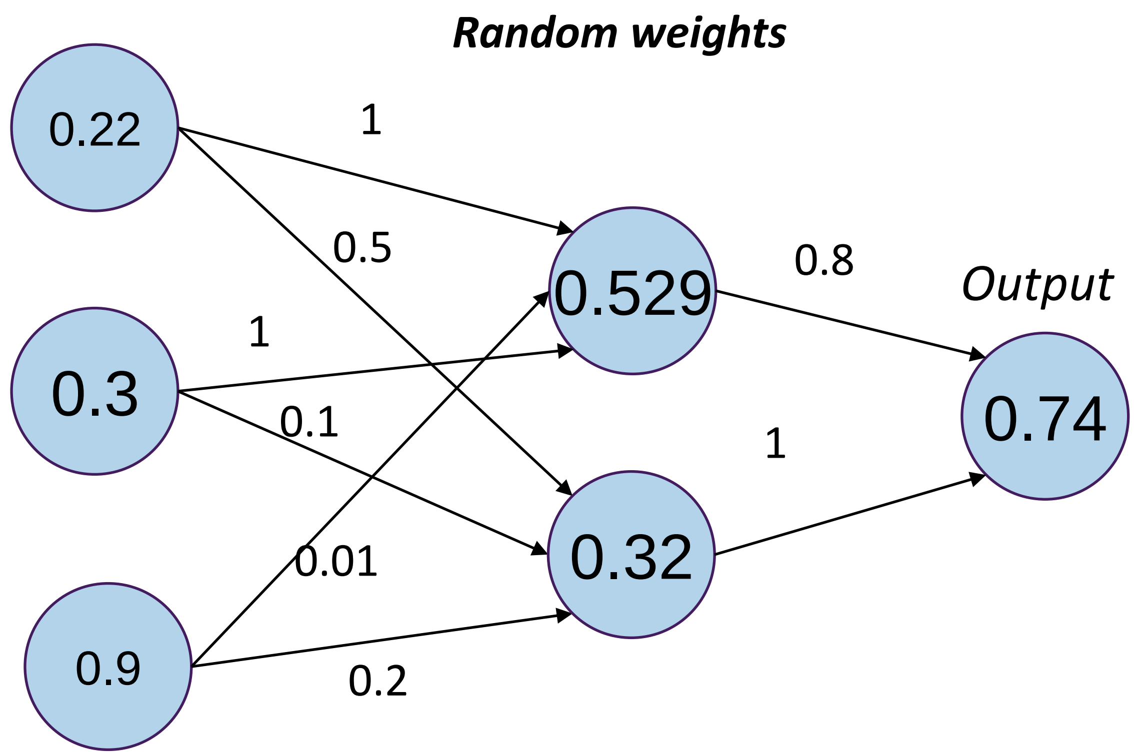
Example No. 2 Artificial Neural network

Inputs

Outputs

X1	X2	X3	Y	
0,22	0,3	0,9	1	**0,74**
0,56	0,8	0,3	0	**1,51**
0,14	0,4	0,66	1	**0,68**
0,78	0,67	0,12	0	**1,64**

"Error" = 3,73



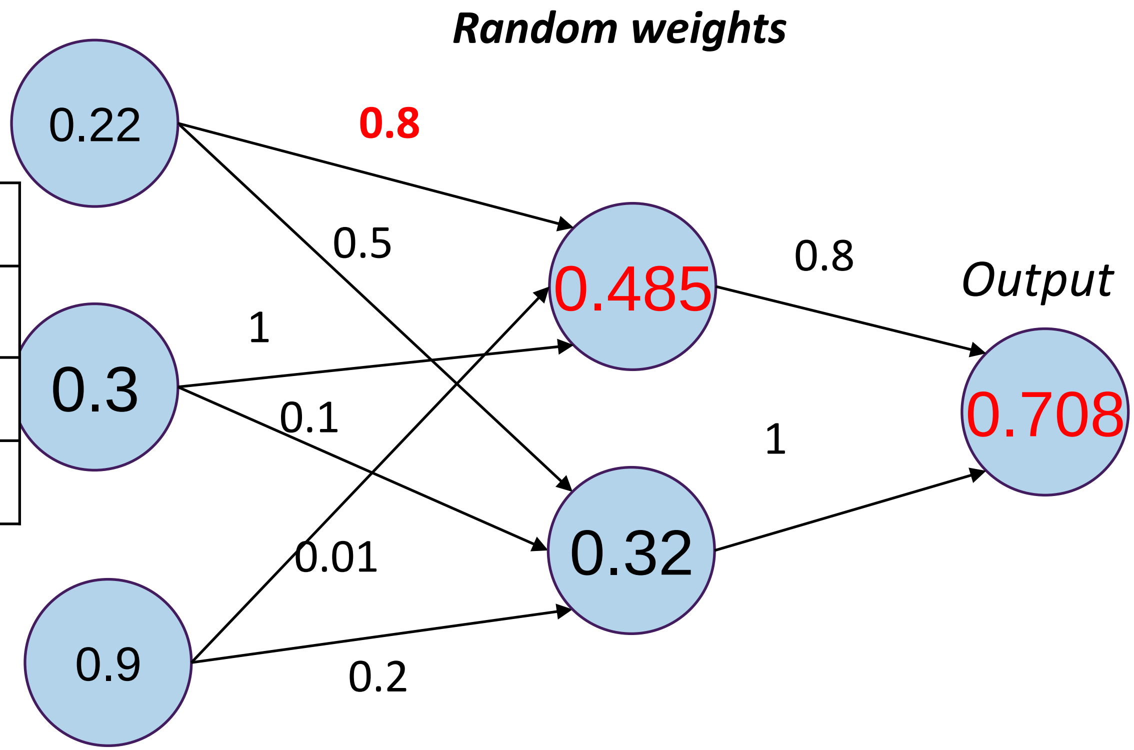
Calculated output .vs real output: $0.74 \neq 1$

Example No. 2 Artificial Neural network

Inputs | Outputs

X1	X2	X3	Y	
0,22	0,3	0,9	1	**0,71**
0,56	0,8	0,3	0	**1,42**
0,14	0,4	0,66	1	**0,66**
0,78	0,67	0,12	0	**1,52**

"Error" = 3,57

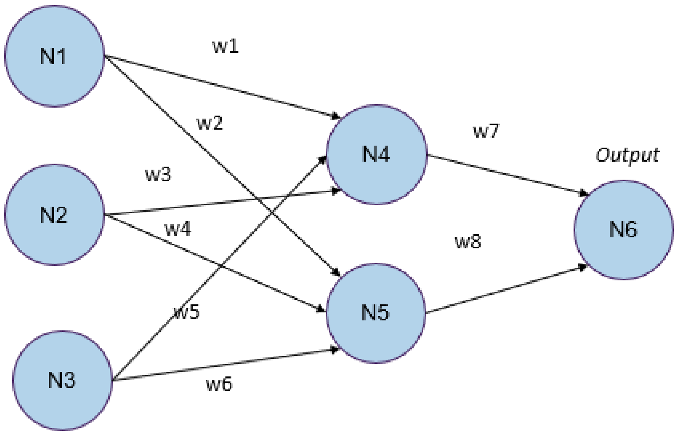


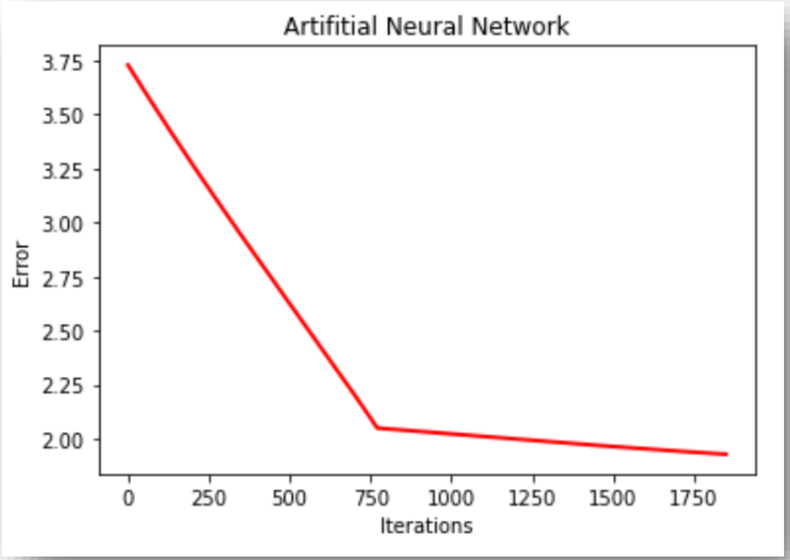
Calculated output .vs real output: $0.74 \neq 1$

Example No. 2 Artificial Neural Network

Code example:

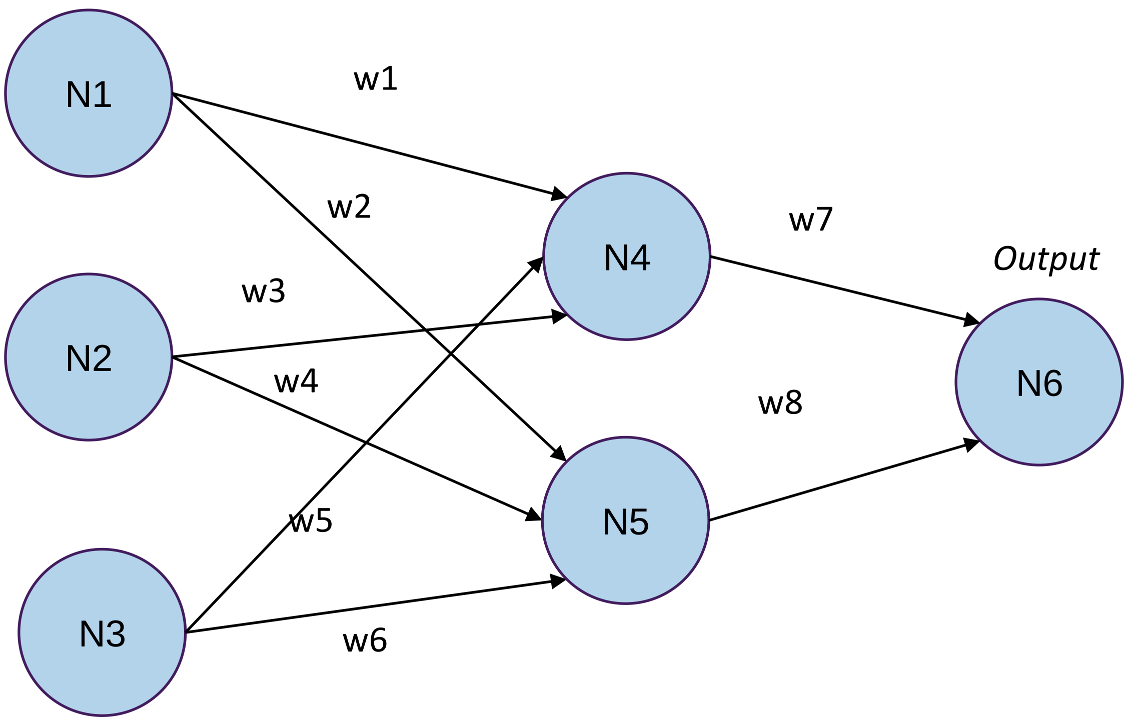
- gdExample.ipynb





To do:

- Apply provided example on continuosDataExample.tsv dataset
- Try different ANN architectures


N1
N2
N3
N4
N5
N6
w1
w2
w3
w4
w5
w6
w7
w8
Output

Questions?

ktu
1922